January 1991

A Hand-Eye-Arm Coordinated System

Sanjay Agrawal
University of Pennsylvania

Ruzena Bajcsy
University of Pennsylvania

R. Vijay Kumar
University of Pennsylvania, kumar@grasp.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-91-05.

A Hand-Eye-Arm Coordinated System

Abstract

In this paper we present the description and experiments with a tightly coupled Hand-Eye-Arm manipulatory system. We explain the philosophy and the motivation for building a tightly coupled system that actually consists of very autonomous modules that communicate with each other via a central coordinator. We describe each of the modules in the system and their interactions with each other. We highlight the need for sensory driven manipulation, and explain how the above system, where the hand is equipped with multiple tactile sensors, is capable of both manipulating unknown objects, but also detecting and complying in the case of collisions. We explain the partition of the control of the system into various closed loops, representing coordination both at the level of gross manipulator motions as well as fine motions. We describe the various modes that the system can work in, as well as some of the experiments that are being currently performed using this system.

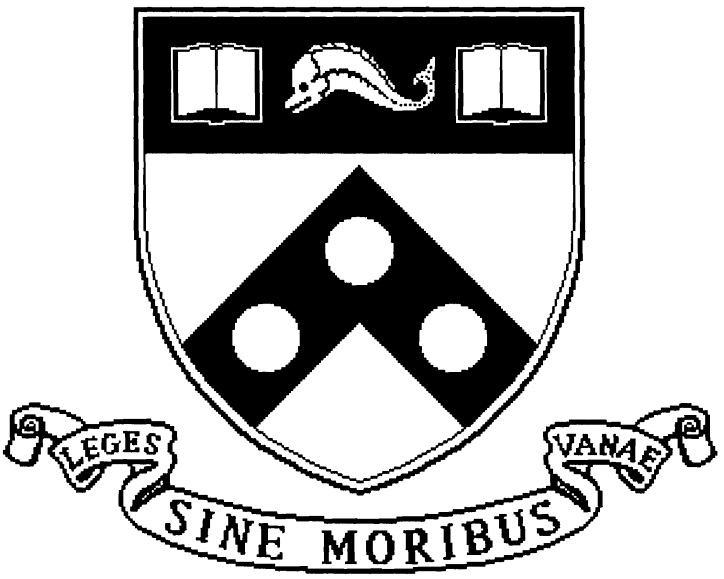
Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-91-05.

A Hand-Eye-Arm Coordinated System

MS-CIS-91-05
GRASP LAB 250

Sanjay Agrawal
Ruzena Bajcsy
Vijay Kumar



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

1991

An Hand-Eye-Arm Coordinated System

MS-CIS-91-05
GRASP LAB 250

Sanjay Agrawal
Ruzena Bajcsy
Vijay Kumar

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

January 1991

An Hand-Eye-Arm Coordinated System

Sanjay Agrawal, Ruzena Bajcsy, Vijay kumar

Department of Computer and Information Science

University of Pennsylvania

January 14, 1991

Abstract

In this paper we present the description and experiments with a tightly coupled Hand-Eye-Arm manipulatory system. We explain the philosophy and the motivation for building a tightly coupled system that actually consists of very autonomous modules that communicate with each other via a central coordinator. We describe each of the modules in the system and their interactions with each other. We highlight the need for sensory driven manipulation, and explain how the above system, where the hand is equipped with multiple tactile sensors, is capable of both manipulating unknown objects, but also detecting and complying in the case of collisions. We explain the partition of the control of the system into various closed loops, representing coordination both at the level of gross manipulator motions as well as fine motions. We describe the various modes that the system can work in, as well as some of the experiments that are being currently performed using this system.

1

[1]Acknowledgements:This work was in part supported by: NSF Grants DMC-8512838, CISE/CDA 88-22719, IRI 89-06770, Navy Grant N0014-88-K-0630, Air Force Grants AFOSR 88-0244, AFOSR 88-0296, Army/DAAL 03-89-C-0031PRI, and DuPont Corporation. We would like to thank Marcos Salganicoff, Michael Chan, Fil Fuma and John Bradley for their contributions.

1 Introduction and Motivation

In trying to perform manipulatory tasks in a unknown or partially modelled world we are quickly reduced to making the world a very restricted domain. We allow for minimal uncertainties, use a *soft* environment to reduce shocks to the devices and limit the motions to simple straight line trajectories. None of these limitations are easily overcome since what we need is a end-effector that is capable of sensing, processing and transmitting any forces or torques back to the arm controller, which then must use this information to reposition the arm [7]. Such a manipulatory system would allow us both create an accurate description of the world [3], as well as be able to function without destroying the environment or the manipulators. As we move the primary domain of robotics away from factory automation to uses such as underwater exploration, nuclear waste cleanup, and garbage disposal, we encounter environments that are not only unknown, but also unpredictable. Such environments would perforce require the ability to of the manipulator to be able to determine the orientation of surfaces, and comply to any changes encountered [2, 4]. As secondary issues that such hazardous environments highlight is the need to protect the sensors on the manipulators as well as the need to have no sharp edges on the manipulators contacting surfaces. Both these issues can be solved by covering the fingertips and palm with rubber skins. Additional benefits of such skins, are increased friction, passive compliance, and soft-contacts.

2 General review of the previous work

Robot systems exist in many varied forms, each focused on a specific aspect of manipulation. There are only a few systems which attempt to grasp objects using sensory information. Hands have been integrated into systems to perform manipulation and grasping tasks for many years though and among these are Geschke's early system to perform robotic manipulation tasks [6], Kuniyoshi et al [10] in building an integrated robotic teaching and learning systems and work at USC in integrating the Belgrade/USC hand into an active perceptual environment [11, 15]. Several research groups working on the MIT/Utah hand have also build integrated systems where the developers of that hand [9, 12] provided a low level control system for the hand and a software environment to utilize the low-level control functions. Many researchers have focused on grasp synthesis using the above hands [8, 13, 14]. Grasp synthesis relies on the ability to arbitrarily position the fingertips at any point within the workspace, a feature which is not present in our system.

Stansfield [18] attempts to first explore and extract the physical and geometric properties of the object,

such as the hardness, weight, size and shape of the object. This is done via built-in so called Exploratory Procedures, that are motoric and sensate procedures used to extract the above mentioned properties. The size and shape is extracted from visual-range sensor data. Stansfield converts these measurements into linguistics labels, such as heavy, soft, large, small etc. which then in turn are used for planning and executing grasp strategies. Allen [1] uses a hand mounted on PUMA560 to extract information from the environment by using tactile sensors mounted on the fingertips of the hand.

2.1 Coordinated System

Roberts [16] shows an algorithm that can compute the joint angles of both the arm and the hand based on positions and outward normals of all the desired fingertip contacts. Such a solution though useful in a completely modelled environment, where positioning accuracy is at a premium, cannot be used in unknown environments. When reference positions in the world are unknown we need tactile capabilities to determine contact points, as well as force sensors to detect contact forces and torques.

None of the above systems address the issue of how one would use the force feedback from the sensors on the hand to drive the motion of the arm. In unknown environments, all collisions need to be detected and complied with, and in addition one would want the ability to trace a surface using purely tactile and force feedback, since the arm and hand would in most cases obscure the field of view. In order for motions of the hand and arm to be coordinated, it is not sufficient to merely be able to provide information about the hand to the arm and vice-versa. Coordination would imply that the two devices should be able to influence the motions of each other at a high bandwidth in order to act in a coupled manner. Vision or model feedback, can provide an initial global description of the environment and can generate gross positional information not sufficient to manipulate objects in an partially environment.

3 System Overview

The system consists of three active control modules. These modules are can interact with each other, via means of a coordinator. The coordinator performs tasks very similar to the human central nervous system, though with nowhere near that complexity. The essential task of the coordinator is to monitor the progress of the current motion commanded from each of the three system modules. There is a front end to each module, that allows the sensory information from each module to be mapped into the current global task framework, and in addition the front end excepts commands for the respective modules, and translates these commands to a format that the module can parse. As a final task, the front end ensures

that all messages are relayed back and forth in a reliable and consistent manner.

3.1 Coordinator

The coordinator itself monitors the current motion, and constantly revises the next motion, for each of the modules. The bases for the revisions of future motions is the current sensory feedback. The coordinator has three motion queues, one for each hand which it uses help build dependencies between future and current motions between modules. The coordinator has three queue managers that monitor the current elements in the queue, while the coordinator can if neccessary modify or disable the queues.

Dependencies between queue elements are introduced when the initial plan is laid out. These dependencies are linked to gross motions. Each element in a module's motion queue is called a gross motion. The fine motions for the hand and arm module computed based on the current mode being used for the motion. These modes determine how the sensory feedback from the hand is interpreted to compute the fine motions of the arm. The coordinator also allows the user to guide the system, The user can freeze the system, change the velocity of the arm, and can restart the system providing it with a new arm configuration. This ability can be useful, if the current arm configuration will not allow objects to be reached or manipulated in cartesian space.

3.2 Hand Module: Software and Hardware

The hand module is run on a PC-AT and linked to the coordinator by a 16 bit parallel bus. The module consists of a graphical user front-end and a communication front-end for the coordinator. Thus the user can type any of the commands via the keyboard that are noramly sent over the parallel bus. The PC has 8 full sized slots that allow for motion controllers, D/A cards, A/D cards for the tactile sensors and encoder decoder cards to be plugged in. The motors of the hand are driven by externally mounted amplifiers. All the data and actuation signals run from the PC to the hand, via a single 112 braided and shielded cable. This allows us to eliminate any heavy actuator packages from having to be mounted on the arm. The Penn Hand [19] itself weighs only 1.5 kgs and is mounted on the robot flange via a quick release mechanism.

3.2.1 Hand Command Description

Since the hand communicates with the coordinator at a low bandwidth as compared with its servo rate, which can be varied between 400 and 700 hertz, the controller in the hand module must be able

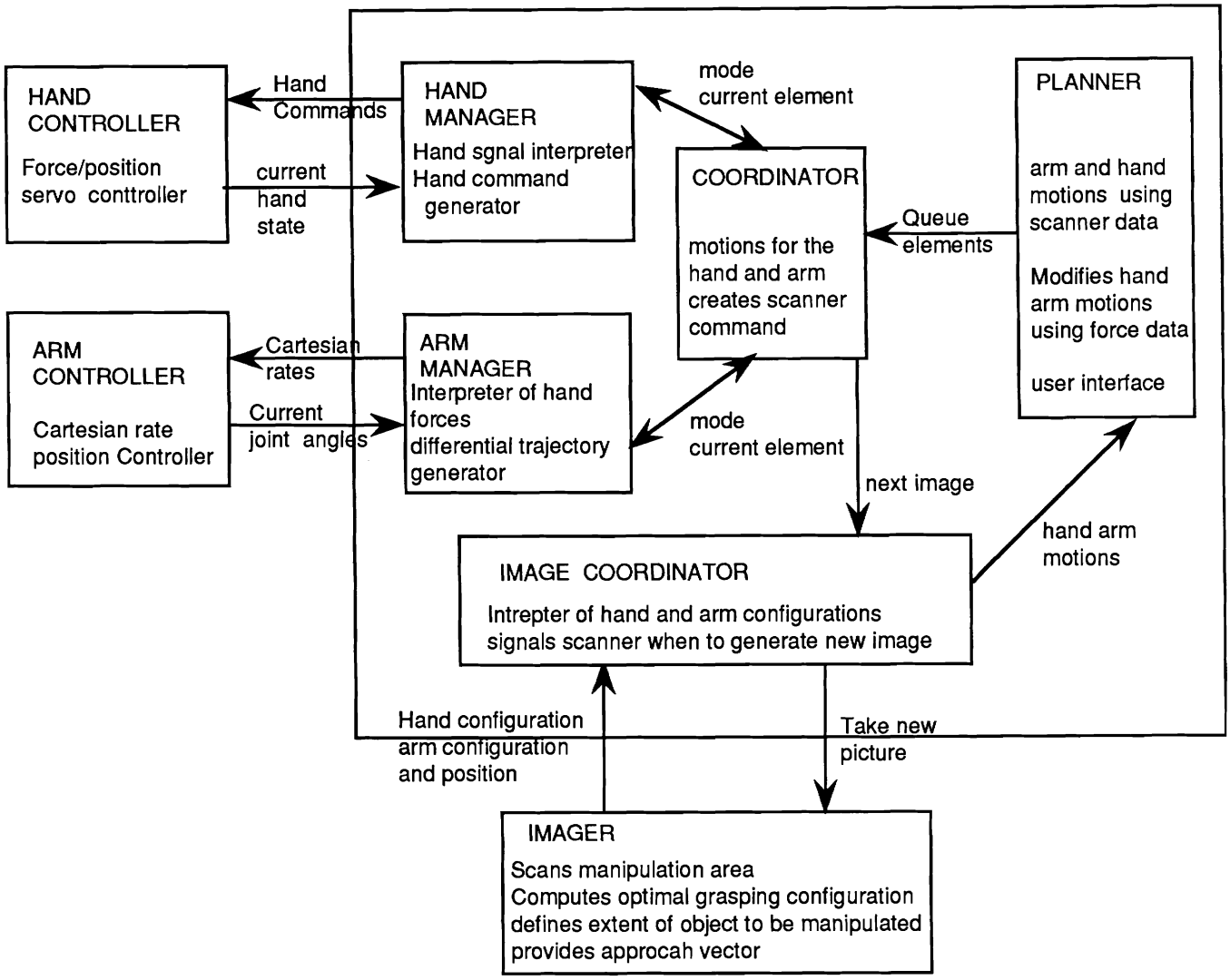


Figure 1: System Command and Data flow

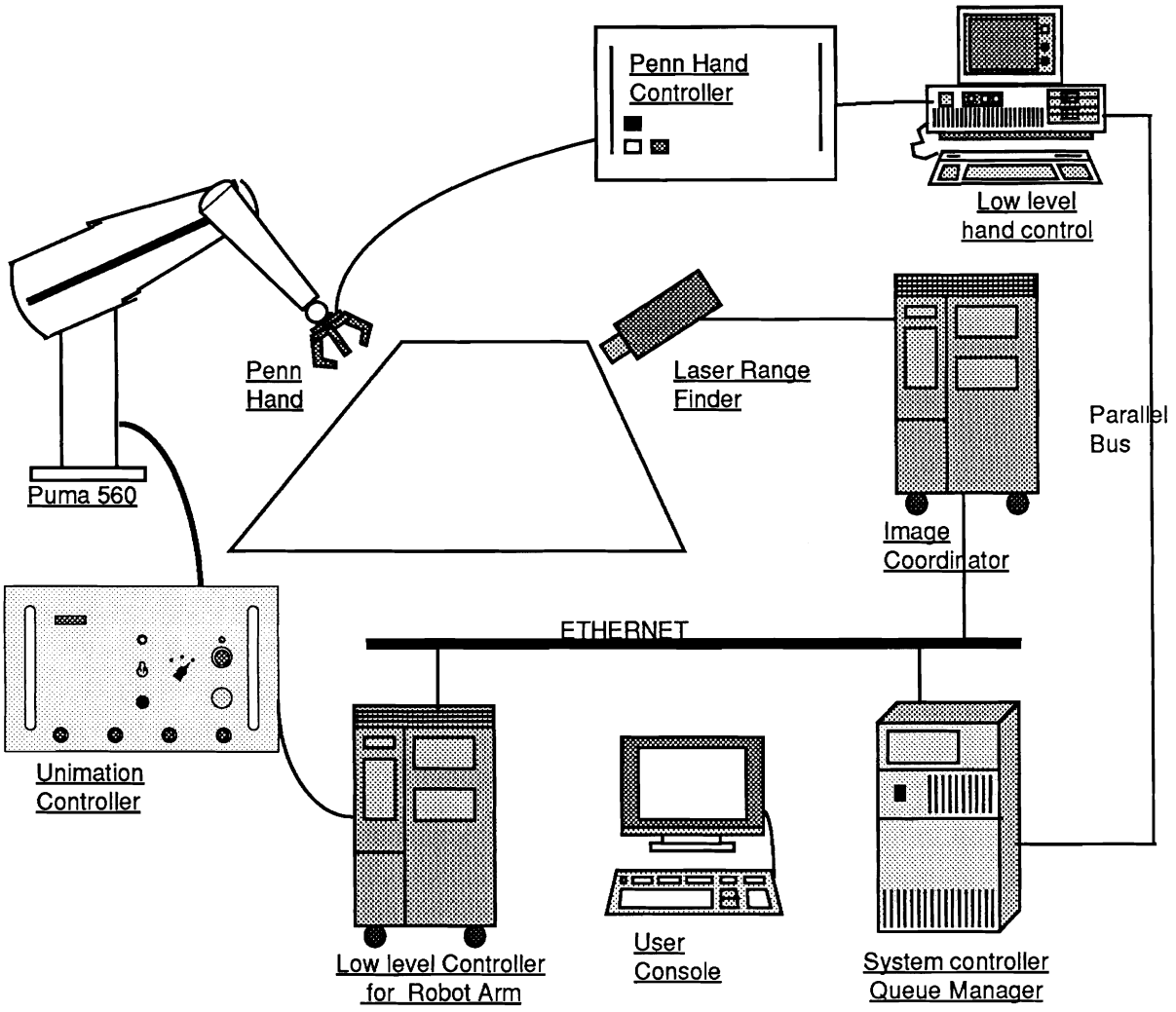


Figure 2: System Hardware Configuration

to autonomously transition from one command to the next. Commands are divided into 5 categories, servo_immediate, servo_future, parameter, calibration and mode. In addition, commands can be addressed one or more joints. Joint dependencies can be specified. Dependency on a joint requires that the previous command for that joint be completed, before the the command sent is valid. Servo commands provide a desired joint angle, a combination of desired sensors to servo on. (Each finger as four spatially distinct sensors) Not all combinations are valid, for instance, one cannot expect forces on both the front and back sensors of the fingers.

Servo_immediate command causes the joints addressed to instantly switch to the given command.

Servo_future command allows specification of other joint commands as dependencies for command execution

Parameter command allows you to specify the finger stiffness and joint velocity.

Calibration commands lets you reset the joint encoders and set the zero value of the sensors.

Mode command allows you to specify the algorithm the force servo uses.

The hand module acknowledges each command, after parsing it, and verifing the fact that it is a legal command. When the module is not parsing a command, it is sending out the the current joint positions, and the forces from the 13 tactile pads. Figure 3 shows the location of these sensors.

3.2.2 Hand Servo Control

The joints of the hand are controlled using a closed PD loop. Since a desired force may also be specified the controller can switch between monitoring the position or the force. Until a force is encountered on the sensor/s that the controller monitors, the controller is in the position servo loop. If the desired position is reached, the current is cutoff since the joints are non_backdrivable. If a force is encountered before or after the desired position is reached, the servo switches to force control, and tries to servo on the desired force. Thus the controller switches between these two modes while the current command is valid.

3.2.3 The Sensorized Penn Hand Description

The Penn Hand was designed to be a medium complexity end-effector, by which we imply, the ability to attain a wide set of hand configurations and grasps while at the same time requiring minimal computational resources and a simple control scheme. The mechanics of the hand and its grasp modes are described in [19].

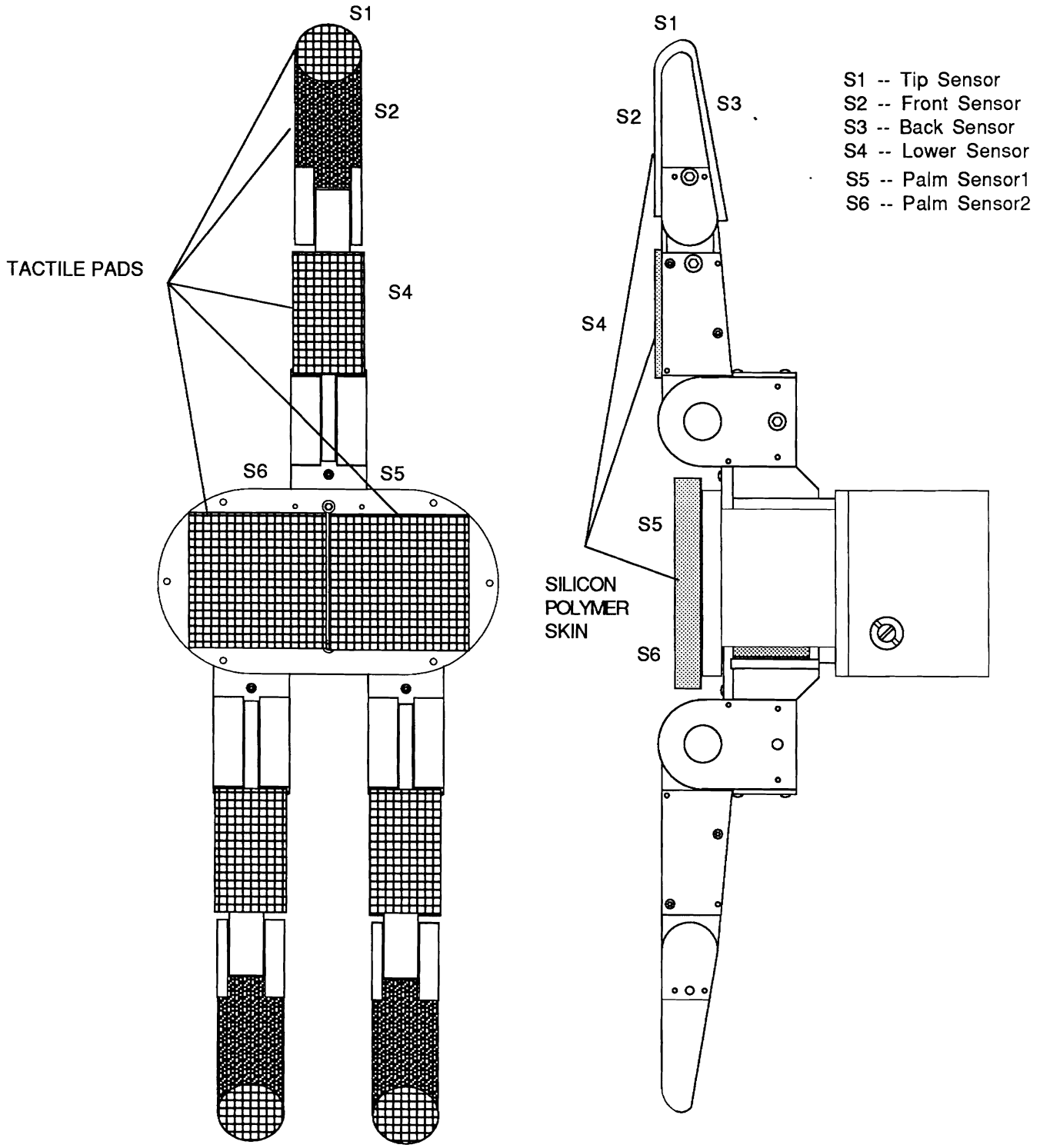


Figure 3: The Sensorized Penn Hand

The hand is primarily a enveloping gripper, with the coupled fingers allowing us to enclose a variety of shapes and sizes. At the same time we have a pinch grip that allows us to pick up objects that are not suitable for enclosure grasps. The other distinctive provision of the Penn hand is the presence of a Palm, around which two of the fingers rotate. The palm which has a compliant skin and touch sensors on the surface, can be used as a support platform, so that the object does not slip when the fingers loosen their grip. This support however only occurs when the palm points upwards.

An important design consideration for the hand was the requirement that one should be able to furnish its surfaces with as many sensors as possible. The palm has two large planer tactile sensors, that cover the entire area of the palm. The fingers, which have two links each, have a total of four sensors on both links together. The lower link has one sensor covering the area facing in towards the palm, and the upper link which is roughly cylindrical in shape, with a hemispherical tip, has three sensors, one facing the palm, one away from the palm, and the last mounted on the tip.

The other important consideration in the design of the palm and fingers is the need for a compliant surface skin. The skin gives us the ability contact stiff surfaces without causing large interactive forces in the now coupled system. In addition the skin can be lubricated, to provide varying friction properties, depending on the task requirements. The skin can also withstand temperature of up to 300^o centigrade, allowing the hand to function in a variety of environments. We can vary both the stiffness and the thickness of the skin to accommodate different requirements.

3.3 Arm Module: Hardware and Software

The arm module runs on a MicroVaxII, which is interfaced to the Unimate robot controller via a parallel line that is tied in to an interrupt line. The operating system is DEVBUS a modified unix kernel, allowing for time critical operations to be performed. The arm module communicates with the coordinator via ethernet using AF_INET stream sockets that allow reliable communication between the coordinator and the controller. The arm controller which accepts differential cartesian rates [5], runs at about 35 Hertz. These cartesian rates are run through the inverse Jacobian to obtain the joint rates which the controller then sends to the *Unimation* box. The arm module relays the current position and orientation of the tool, as well as the current configuration over to the coordinator. If the module does receive a new differential cartesian change from the coordinator within a fixed number of cycles, the controller will be asked to cut the joint rates to zero, till the next rate is sent down.

3.4 The Image Coordinator

The Image Coordinator runs on a MicroVaxII, to which is hooked up to a laser scanner. Here again communication between the coordinator and the imager is done over ethernet using AF_INET stream sockets. When the imager receives a command to scan a new picture, it takes an image and if it finds an object attempts to fit a *Superquadric* [17] on it.

The Superquadric model is then analyzed to determine the most optimal means of grasping it. There are three parameters that are optimized, distance arm has to travel, number of degrees of freedom the object could be constrained in, and distance of the center of the palm from the centroid of the object model. If a successful grasp is found, an approach vector is determined based on the additional constraint that one cannot easily change the wrist configuration of a PUMA560, since that would imply going through a singularity since we move in cartesian space. If a successful approach vector is determined, an approach position, a expected contact position and a hand configuration is computed and sent along with the object dimensions to the coordinator.

3.5 Motion Planning

Planning is the high level process that interacts primarily between the queue managers, the imager and the user. The planner can either have the user provide a task framework, or use a predefined framework to create the motion queues. The planner built into the system currently can construct two possible task frameworks autonomously.

3.5.1 Grasping single objects

The first framework is setup by using the feedback from the visual sensor, to create motion queues for the arm and hand, specifying among other points, an approach point, a expected contact point and a release position. The planner looks at the list of positions sent by the image coordinator, and determines the point at which the hand needs to start preshaping so as to be able arrive at the approach desired approach position in the desired hand configuration. The hand should preshape as late as possible, to allow the hand to remain in what we call the *comply* position as long as possible. The *comply* position is particular set of finger tip positions that best allow the system to track or contour surfaces. This position is explained later on in the section on algorithms.

Thus while moving to the approach position, the arm moves along the shortest possible path towards the approach point, until it hits an obstacle or reaches the point. If an obstacle is encountered the system

computes the surface normal, by bringing all three fingertips into contact with the surface. Once the surface normal is detected the arm maintains a trajectory towards the approach point either complying with the surface if the surface prevents a shortest path, or moving in free space. Once the hand is in free space the arm reorients itself to the desired approach vector.

When the approach point is reached the arm moves the arm along a straightline, until one of the fingers or the palm makes contact with the object. If a finger makes contact, the arm can be moved along the object, till the fingers now enclose the object. If the palm contacts the object then the fingers simply close on the object. If no contact has been made once the contact point is reached the system asks the user to provide the next task framework. Once the object is enclosed the hand maintains the desired force on the object. The desired force is computed by the force controller on the hand module, which can modify the desired forces if the current forces are not sufficient to hold the object firmly. The arm then moves the now firmly enclosed object through the next set of desired positions, before releasing it in the desired position.

We have not as of yet addressed the issue of how the system should respond to the situation when obstructions in reaching the goal, prevent the hand from reaching the object.

3.5.2 Pulling at single degree of freedom objects

The second task framework is also setup using feedback from the visual scanner. The scanner provides the planner with an object location and its degree of freedom. We assume that the object has a prismatic degree of freedom, and could possible become free of environment. The planner once again creates motions queues for the hand and arm, then monitors the motions till the object is reached, the same strategy as in framework one. Once the object is grasped, the arm moves along the degree of freedom, monitoring the change of forces in the sensors S2 and S4. When the forces at these sensors increase by a significant amount, the hand releases the object. The assumption made here, is that the object has reached the limit of its motion. After a fixed amount of travel, if the forces stay constant, the arm checks to see if the object has come free, by testing an orthogonal degree of freedom. If no large interaction forces are encountered, then the arm is free to move to the release position, else the object is released at that point.

3.6 System Modes

In the completion of a task, the hand-arm system iterates through several global modes of operations. These modes comprise of set of modes for the hand, arm and scanner. Each mode for the arm, requires

the controller to interpret the force feedback from the hand in a distinct manner. The hand modes are based on the current function the hand is performing. The mode for the hand determines under what conditions the hand executes the next command. Transition from one mode to occurs via sequential queues that are set up for both the hand and the arm. The transition between modes is decided by the respective queue managers. The queues incorporate dependencies between the desired motions of the hand and the arm, as well as the requirement for new images to be obtained via the laser range scanner, these dependencies are managed by the coordinator.

3.7 Software design and Algorithms

The software is designed with two fundamental requirements in mind. One is the need for the system to operate independently of the other manipulators and sensors, and two the ability to integrate other sensors and manipulators, without modifying any of the existing modules. The only module in the system that must know about all the sensors and manipulators in the system, is the central coordinating module, which must initialize and startup each module. Since the communication package is a standard unix interface, any external program can communicate with the system once the coordinator is alerted to its presence.

3.7.1 Surface Normal Extraction

This algorithm reads the forces on the three fingertip sensors and uses a predetremined stiffness constant to compute a displacement for the reading. This displacement subtracted from the forward kinematics computation for the finger position for each of the fingers, gives a plane with reference to the tool frame of the robot. This can information can then provide the system with the surface normal of the plane.

3.7.2 Complying with the surface

In order to comply with the surface, the system attempts to find the surface normal, and then orient the approach vector of the tool frame along the same direction. This ensures that the hand is aligned to the surface normal in the static case. In the dynamic stage, the fingertips tend to have varying forces, which change faster than the servo rate of the arm. In order to ensure that the arm does not get unstable, the fingers of the hand comply to reduce the force, by increasing the apperture. The force are still transmitted to the arm, which can now comply at a much lower rate, and with a much smaller gain. The fingers return to the fixed *comply* position as soon as the arm has moved sufficiently to allow them back again.

4 Conclusion

What we demonstrate in this system is the need for integration of various sensory and manipulatory modules that can function in a coordinated manner. In addition we claim, that unless the system has the ability to integrate the information obtained from the various sensors, and use it to control the motions of all the manipulators, albeit in different strategies, we will not be able to work in environments that are unpredictable and not completely modelled.

References

[1] P. Allen, P. Michelman, and K.S. Roberts. An integrated system for dextrous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 612–61, Scottsdale AZ, 1989.

[2] R. Bajcsy. What can we learn from one-finger experiments. In M. Brady and R. Paul, editors, *Robotics Research, First International Symposium*, chapter , pages 509–529, MIT press, 19.

[3] D. Brock and S. Chiu. *Environment Perception of an Articulated Robot Hand Using Contact Sensors.* Technical Report , MIT, Department of Mechanical Engineering, 1985.

[4] G. Buttazo, P. Dario, and R. Bajcsy. Finger based explorations. *SPIE*, 726:, October 1986.

[5] Peter I. Corke and Richard Paul. *Video-Rate Visual Servoing for Robots.* Technical Report MS-CIS-89-33, GRASP Lab, University of Pennsylvania, 1989.

[6] C.C. Geschke. A system for programming and controlling sensor-based robot manipulators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):1–7, January 1983.

[7] K.Y. Goldberg and R. Bajcsy. Active touch and robot perception. *Cognition and Brain Theory*, II(2):, 1984.

[8] J. Hong, G. Lafferriere, B. Mishra, and X. Tan. Fine manipulation with multifingered hands. In *IEEE International Conference on Robotics and Automation*, pages 1568 – 1574, Cincinnati OH, 1990.

[9] S. C. Jacobsen, J. E. Wood, D. F. Knutti, and K. B. Biggers. The utah/mit dextrous hand: work in progress. *The International Journal of Robotics Research*, 4(3):21–50, 1985.

[10] Y. Kuniyoshi, H. Inoue, and M. Inaba. Design and implementation of a system that generates assembly. In *International Workshop on Intelligent Robots and Systems IROS*, pages 567–574, 1990.

[11] H. Liu, T. Iberall, and G.A. Bekey. The multi-dimensional quality of tasks requirements for dextrous robot hand control. In *IEEE International Conference on Robotics and Automation*, pages 452–457, Scottsdale AZ, 1989.

[12] S. Narasimhan, D.M. Siegel, J.M. Hollerbach, K.Biggers, and G.E. Gerphiede. Implementation of control methodologies on the computational architecture for the utah/mit hand. In *IEEE International Conference on Robotics and Automation*, pages 1884–1889, San Francisco CA, 1986.

[13] T. Nguyen and H. Stephanou. A topological algorithm for continuous grasp planning. In *IEEE International Conference on Robotics and Automation*, pages 670 – 675, Cincinnati OH, 1990.

[14] Y. Park and G. Starr. Optimal grasping using a multifingered robot hand. In *IEEE International Conference on Robotics and Automation*, pages 689 – 69, Cincinnati OH, 1990.

[15] K. Rao, G. Medioni, H. Liu, and G.A. Bekey. Robot hand-eye coordination:shape description and grasping. In *IEEE International Conference on Robotics and Automation*, pages 407–411, Philadelphia PA, 1988.

[16] K.S. Roberts. Coordinating a robot arm and multi-fingered hand using the quaternion representation. In *IEEE International Conference on Robotics and Automation*, pages 1252 – 1257, Cincinnati OH, 1990.

[17] Franc Solina. *Shape Recovery and Segmentation with Deformable Part Models.* PhD thesis, University of Pennsylvania, 1987.

[18] S.A. Stansfield. *Haptic Perception with an Articulated Sensate Robot Hand.* Technical Report SAND90-0085.UC-406, Sandia National Laboratory, 1990.

[19] Nathan Ulrich, Richard Paul, and Ruzena Bajcsy. A medium complexity compliant end-effector. In *IEEE International Conference on Robotics and Automation*, pages 434–437, Philadelphia PA, 1988.